大范围动态海浪实时渲染

王 纲[1], 季振洲[1], 张泽旭[2]

(1. 哈尔滨工业大学 计算机科学与技术学院, 150001 哈尔滨, gmkwangg@gmail.com;
2. 哈尔滨工业大学 深空探测基础研究中心,150001 哈尔滨)

摘 要: 为实现大范围动态海浪的实时渲染提出了一种高效方法.首先提出一种基于LOD技术的新型网格模型,采用Gerstner模型在GPU(图形处理单元)上实现了海浪的动态模拟和Choppy波的模拟,给出一种凹凸纹理贴图的方法对较远处的海浪渲染实现加速,并利用动画凹凸纹理改善海浪渲染效果;其次通过立方体纹理实现了对天空、太阳的反射效果,运用Phong光照模型实现了海浪的光照并实现了菲涅尔反射;最后采用动画纹理模拟了海浪中的泡沫,并实现了云在海面上投射的阴影等特殊效果.实验结果表明,该方法可以对90 km范围的海浪进行实时交互渲染,并已成功应用于某飞行模拟器中.

关键词: 动态海浪;实时渲染;Gerstner模型;纹理动画
中图分类号: TP391 **文献标志码:** A **文章编号:** 0367-6234(2012)03-0059-05

Real-time rendering of large scale dynamic ocean waves

WANG Gang[1], JI Zhen-zhou[1], ZHANG Ze-xu[2]

(1. School of Computer Science and Technology, Harbin Institute of Technology, 150001 Harbin, China, gmkwangg@gmail.com;
2. Deep Space Exploration Research Center, Harbin Institute of Technology, 150001 Harbin, China)

Abstract: In order to render large scale ocean waves in real-time, an effective method is presented. Firstly, a novel geometry model based on LOD (Level of Detail) is used as the ocean surface, and the Gerstner wave is applied on GPU to simulate dynamic ocean wave and choppy wave. A bump mapping texture is utilized on areas far from the eyes for improving rendering performance, and the animated bump mapping textures are used to enhance the rendering effect of ocean surface. Secondly, the sun and sky reflections are achieved via cube mapping texture. Phong light and Fresnel reflection are discussed and applied. Special effects such as cloud shadow and whitecaps based on texture animation are implemented. Experimental results show that the method can be used to render a 90 km ocean scene interactively in real-time and has been applied to a flight simulator successfully.

Key words: dynamic ocean wave; real-time rendering; Gerstner wave; texture animation

地球表面约70%的区域被海洋所覆盖,相比较其他自然景物的仿真,海浪场景的绘制在飞行仿真、水面舰船仿真应用中显得更为重要.多年来,有大量针对海浪的研究,Mihalef将渲染效果的尺度分成了3类,所用的方法归结起来可以分为两大类:基于物理模型的方法和基于构造的方法[1].基于物理模型的方法一般是从Navier-Stokes方程组着手,试图通过求解该方程组来实现对海浪的模拟;基于构造的方法通过构造网格或基于粒子的方式,对海浪进行模拟[2-4].但是,直到Stam提出了无条件稳定的算法,才解决了算法的稳定性问题.基于物理模型的方法更适合于流体倾倒、喷泉等小范围、小规模的场景,或是用于电影、动画等不要求实时性的场合.由于Navier-Stokes方程组的求解极为复杂,即便现代GPU的计算性能已达到TFLOPS量级,仍然无法通过这类方法交互实时地实现大范围动态海浪的渲染.

在实时交互性应用中,以往采用的是基于构

收稿日期: 2011-03-09.
基金项目: 航天创新基金资助项目(CASC200902-4);深空探测着陆与返回控制技术国防重点学科实验室开放基金资助项目(HIT.KLOF.2011.077).
作者简介: 王 纲(1967—),男,博士研究生,高级工程师;
季振洲(1965—),男,教授,博士生导师;
张泽旭(1971—),男,副教授,博士生导师.

造的方法,通过数学函数的方法构造出海浪的外形. 基于几何模型的方法是常用的构造方法,通过采用函数曲线等来模拟海浪的几何形状,该方法波形容易控制,速度较快,其不足之处在于波形过于规则,真实感差,已经难以满足目前对海浪模拟真实性的要求[5-6]. 基于统计和谱的方法是目前研究的主要方向,该方法依据海洋统计和海洋观测的经验模型,采用多个正弦曲线或快速傅立叶变换合成符合海浪谱分布的海浪[7-9]. 这类方法在现代 GPU 上也很容易实现,从而为实现实时交互提供了可能. 现代 GPU 为基于 GPU 的海浪仿真提供了良好的平台和巨大的灵活性. Mitchell[10]在早期的可编程 GPU 上实现了一种多尺度 FFT 的海浪模拟方法,该方法在比较粗糙的网格上通过 FFT 生成了低精度的海浪高度图来模拟海浪的动态,通过精细的采样网格计算海浪的法线图,实现海浪的光照效果,并对航迹以及浅水区海浪的衰减现象进行了模拟. Kryachko[11]利用基于 GPU 顶点纹理位移映射技术和射线同心圆网格模型,通过高度图和法线图叠加的方法实现了海浪的真实感绘制. Finch[12]则在 GPU 上利用网格的几何波动与动态法向量扰动来绘制大规模海浪场景. Isidoro 等[13]提出了一种完全基于 GPU 的实时海浪绘制方法,在顶点着色器中用 4 个正弦波叠加计算海浪的几何模型,在像素着色器中实时计算海浪的光照效果,较好的实现了大范围海浪的实时渲染. 李起成等[14]基于 GPU 在考虑动态天空环境下实现海浪实时渲染. Bruneton[15]采用 FFT 在 GPU 上实现了海浪的渲染,并采用从几何模型到 BRDF 的无缝变换实现了海浪的光照.

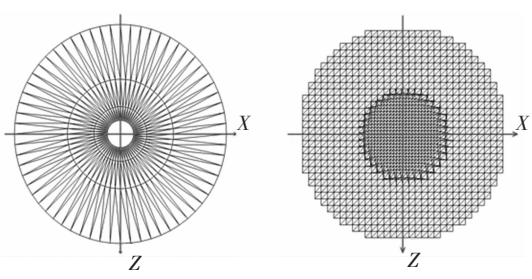
本文采用统计和谱的方法,研究了利用 GPU 实现大范围复杂场景下动态海浪的实时渲染技术. 首先提出了一种新的基于 LOD 技术的网格模型来模拟海平面,在高等级 LOD 上采用基于 GPU 的 Gerstner 模型直接计算海浪的高度并计算其法向量,在低等级的 LOD 上计算海浪的凹凸纹理图;最后模拟海面的光照及一些特殊效果,包括细碎海浪的模拟、海面对天空与太阳的反射效果、泡沫以及云层在海面上的阴影等.

1 网格模型

在进行海浪仿真时,一般采用各类网格模型来模拟海平面. 网格模型的拓扑结构以及网格间距直接影响海浪渲染的实时性和真实感. 过小的网格间距势必会增加计算的负荷,影响海浪渲染的实时性;反之,过大的间距将降低海浪的精度,影响模拟的真实感. 网格的间距通常取决于模拟的海况,低海况时由于海浪主要是以小振幅高频波的形式出现,因而网格间距要小一些,高海况时海浪以大振幅低频波的形式出现,可以采用较大的网格间距.

最常用的一种网格模型是由 $M \times N$ 个等间距顶点组成的,这种模型由于计算负荷过大难以直接应用于大范围海浪的渲染,主要用于计算高度图、法线图或技术演示等. 一种改进的方法是在这种网格的外围增加大尺度网格作为静态海面,而 $M \times N$ 个等间距顶点组成的网格作为动态网格,并以视点位置作为网格的中心,且只对动态网格进行计算. 采用这种模型的一个主要问题是动态海面与静态海面接合部的模拟效果会出现跳变,真实感较差. Hinsinger 等[16]提出了一种非均匀分布的海平面网格模型,这种模型将可见范围内的海面细分为 $M \times N$ 个矩形网格,并将其按逆透视投影到海平面上,从而获得不均匀的 $M \times N$ 个网格. Johanson[17]则提出了投影网格模型,在投影空间建立一个与视线垂直的均匀网格平面,再将该平面投影到世界空间的海平面上. Kryachko 采用了一种与视点相关的射线同心圆网格模型,视点位于同心圆的中心,其细分特点是离视点越近则提供越多的细节,绘制的海平面呈现出比较好的真实感,同时由于远处的网格细节较少,保证了渲染的实时性.

同心圆网格模型的一个明显不足是只能通过高度图来实现海浪的渲染,而在视点附近,这种方法的真实感不够高. 本文提出了一种新的基于 LOD 的网格模型. 该模型借鉴射线同心圆网格模型和动态、静态网格模型,在视点附近,采用两级细分圆形网格构成两级 LOD,网格中顶点的间距由模拟的海况确定,两级 LOD 以外的网格部分,采用同心圆网格,如图 1 所示.



(a) 静态网格 (b) 动态网格

图 1 海平面同心圆网格模型

设在三维世界坐标 XYZ 中,模拟的海平面位于 XZ 平面,Y 轴垂直向上,视点位置在 $S(S_x, S_y, S_z)$ 处,海平面的可见距离为 R_{max},过视点位置作海平面的垂线,得到交点 S;以 S 为中心,首先构造两级 LOD 圆形网格作为动态网格,对于网格中某一行或某一列顶点的个数,可以通过图形学中圆

的扫描转换算法求取,由于 LOD_1 中的网格间距为 LOD_0 中的网格间距的 2 倍,因此在求取 LOD_0 网格顶点数时,应选择偶数作为扫描转换算法中的半径,并调整相临的奇数行和偶数行的顶点数使之相等,保证两级 LOD 之间不形成交叉,两级 LOD 间的 T 形点可通过修改 LOD_0 中奇数行(列)顶点使之与偶数行(列)顶点重合来消除. 同心圆部分以 S 为圆心,以递增的距离为半径做 N 个同心圆,对第 $i(1 \leqslant i \leqslant N)$ 个同心圆进行 M 等分,各等分点即为同心圆网格模型的网格顶点,各同心圆半径的计算公式为

$$\begin{cases} R_i = 2R_{i-1}, i = 1,2,\cdots,N-1; \\ R_i = R_{\max}, i = N. \end{cases}$$

式中: R_i 为第 i 个同心圆的半径; $R_{\max}$ 为海平面的可见距离.

设第 i 个同心圆上第 j 个网格顶点的坐标为 (x,y,z),其计算公式为

$$\begin{cases} x = R_i\cos(2\pi/M); \\ y = 0; \\ z = R_i\sin(2\pi/M), (i = 1,2,\cdots,N, j = 0,1,\cdots,M). \end{cases}$$

同射线同心圆模型类似,这种网格模型按照顶点与视点的距离进行细分的网格,在视点附近的海面网格划分的比较细,渲染的海浪效果具有较好的真实感;而距离视点比较远的海面采用了较粗的网格,细节得到了必要的简化,从而保证了海浪渲染的实时性,相比于其他网格模型,本文提出的网格模型具有以下特点:

1) 在视点附近采用两级细分圆形网格,为海浪的渲染提供了必要的细节,渲染结果真实感好.

2) 在以视点为中心的各个方向上,构成网格的顶点数基本一致,保证了渲染帧数的稳定.

3) 网格的顶点以视点为中心,随着与视点距离的增加,其间距不断加大. 量测偏导数,以对待估参数进行确定.

与射线同心圆相比,本文提出的网格模型增加了一定数量的顶点,但是,由于仅增加极少的渲染批次[18],因而增加的顶点对渲染效率影响不大.

2　海浪模拟

采用逐顶点计算与凹凸纹理贴图相结合的方法模拟海浪,在细分圆形网格上采用逐顶点计算,在同心圆网格上采用凹凸纹理贴图法. 逐顶点计算可以保证海浪渲染效果的真实感,凹凸纹理贴图法可以提高海浪渲染的效率.

2.1　基于 Gerstner 模型的海浪高度计算

Gerstner 模型最早是由 Fournier 和 Reeves 引入到计算机图形学中用于对海浪的模拟,其理论基础是模拟海平面上的某点在海平面上的运动,当水波经过海平面时,该点将随着水波的运动产生周期性运动. 设该点在未受扰动的海平面上的坐标为 $\boldsymbol{x}_0(x_0,z_0)$,未受扰动时该点的高度 $y_0 = 0$,则在时刻 t,当一个振幅为 A 的水波经过时,该点的运动为

$$\begin{cases} \boldsymbol{x} = \boldsymbol{x}_0 - \dfrac{\boldsymbol{k}}{k}A\sin(\boldsymbol{k}\cdot\boldsymbol{x}_0 - \omega t), \\ y_0 = A\cos(\boldsymbol{k}\cdot\boldsymbol{x}_0 - \omega t). \end{cases} \tag{1}$$

式中向量 $\boldsymbol{k}$ 称之为波向量,是指向水波的运动方向的平面向量,其长度为 k. k 与水波波长 λ 的关系为

$$k = 2\pi/\lambda. \tag{2}$$

式中 ω 为水波的角频率. 在较深的水中,$\omega = \sqrt{gk}$.

按照式(1) ~ (2)生成的海浪过于简单,一个重要的原因是因为式(1) ~ (2)中在水平和垂直方向上只有一个正弦波. 正如上述讨论的那样,可以采用一系列正弦波叠加的方法. 设给定第 i 个波的波向量为 $\boldsymbol{k}_i$,振幅为 A_i,频率为 ω_i,相位为 φ_i,则 t 时刻的海浪运动为

$$\begin{cases} \boldsymbol{x} = \boldsymbol{x}_0 - \sum\limits_{i=0}^{N}\dfrac{\boldsymbol{k}_i}{k_i}A_i\sin(\boldsymbol{k}_i\cdot\boldsymbol{x}_0 - \omega_i t + \varphi_i), \\ y_0 = \sum\limits_{i=0}^{N}A_i\cos(\boldsymbol{k}_i\cdot\boldsymbol{x}_0 - \omega_i t + \varphi_i). \end{cases}$$

海浪的波长和振幅可由 Pierson-Moskowitz 海浪谱计算得到.

2.2　法向量计算

法向量计算是进行光照计算的基础,在任意时刻,海平面上的某点的位置和高度是已知的,因此该点所处平面的方向是可以计算出来的, 因而其副法线和切向量就是该点在 x 和 z 方向的偏导数. 设 t 时刻平面上一点 $\boldsymbol{x}_0(x_0,z_0)$,在三维空间中的坐标 $\boldsymbol{P}$ 为

$$\boldsymbol{P}(x_0,t) = (\boldsymbol{x}_0, \boldsymbol{H}(\boldsymbol{x}_0,t), z_0).$$

其副法线即为 x 方向的偏导数为

$$\boldsymbol{B}(x_0) = \left(\frac{\partial x}{\partial x}, \frac{\partial \boldsymbol{H}(\boldsymbol{x}_0,t)}{\partial x}, \frac{\partial z}{\partial x}\right) = \left(1, \frac{\partial \boldsymbol{H}(\boldsymbol{x}_0,t)}{\partial x}, 0\right).$$

其切线为 z 方向的偏导数为

$$\boldsymbol{T}(\boldsymbol{x}_0) = \left(\frac{\partial x}{\partial z}, \frac{\partial \boldsymbol{H}(\boldsymbol{x}_0,t)}{\partial z}, \frac{\partial z}{\partial z}\right) = \left(0, \frac{\partial \boldsymbol{H}(\boldsymbol{x}_0,t)}{\partial z}, 1\right).$$

所求的法线为

$$\boldsymbol{N} = \boldsymbol{B} \times \boldsymbol{T}.$$

2.3　Choppy 波

在比较平静的海面上,海浪的波峰和波谷都

是比较圆滑的,对于有一定的风作用的海面,波浪的形式为 Choppy 波,其波峰比较尖锐而波谷则比较平坦,式(1)通过在水平方向上移动网格的顶点,从而较好的模拟了 Choppy 波. 式(1)中 kA 为水波在水平方向运动的振幅,在不考虑具体尺寸的情况下,当 $kA \leqslant 1$ 时,随着其值的增加,波峰越来越尖锐,而当 $kA > 1$ 时,将会出现渲染错误.

2.4 渲染效果改进

Gerstner 模型采用有限数量的正弦波对海浪进行模拟,其渲染效果与基于 FFT 的方法相比缺少足够的细节,真实感有一定的差距,如图 2(a)所示,基本看不到细节. 本文通过离线渲染的方式得到动画凹凸纹理,为 Gerstner 模型增加渲染细节,使得渲染的真实感得到了大幅度提高,如图 2(b)所示.

2.5 GPU 实现

首先采用等间距网格,利用 Gerstner 模型计算各顶点的高度,并逐点计算其法向量,将结果渲染到一张凹凸纹理,用于对距离视点较远处的静态网格渲染. 渲染基于 LOD 的圆形网格方法与其类似,由 Gerstner 模型计算顶点的高度、水平偏移量及其法向量,根据计算结果确定动画凹凸纹理的坐标,同心圆网络模型通过对几何模型的顶点信息计算实现参数化并进行纹理映射. 本文采用 Cg 语言实现了上述算法,本文的算法也可以通过 CUDA 或 OpenCL 实现,渲染结果如图 2(b)所示.

3 光照及特殊效果

3.1 海面的光照效果

图 2(b)的效果虽然具有丰富的细节,但依然不够真实,主要是因为光照尚未实现. 海平面对光线具有反射和折射两种作用,海平面的颜色主要是这两者作用的结果.

海平面反射的主要是天空和海平面以上的景物,天空包括云、太阳等,景物主要由地形及建筑物等,其反射效果可通过不同的方法实现. 对天空的反射可以通过环境立方体纹理的方式实现;对海平面以上的景物可以采用平面反射法实现,将海平面以上的景物渲染成纹理,在 GPU 中求取纹理投影坐标,从而实现反射效果. 采用 Phong 模型对太阳等光源的光照进行模拟,如图 2(c)所示.

折射主要是对海平面以下的景物进行的,其实现方法与实现反射的方法类似.

Fresnel 反射是海平面光照计算的重要组成部分, 海平面光照的反射率 R 和折射率 T 满足 $R + T = 1$. 根据 Fresnel 公式

$$R = \frac{1}{2}\left(\frac{\sin^2(\theta_t - \theta_i)}{\sin^2(\theta_t + \theta_i)} + \frac{\tan^2(\theta_t - \theta_i)}{\tan^2(\theta_t + \theta_i)}\right).$$

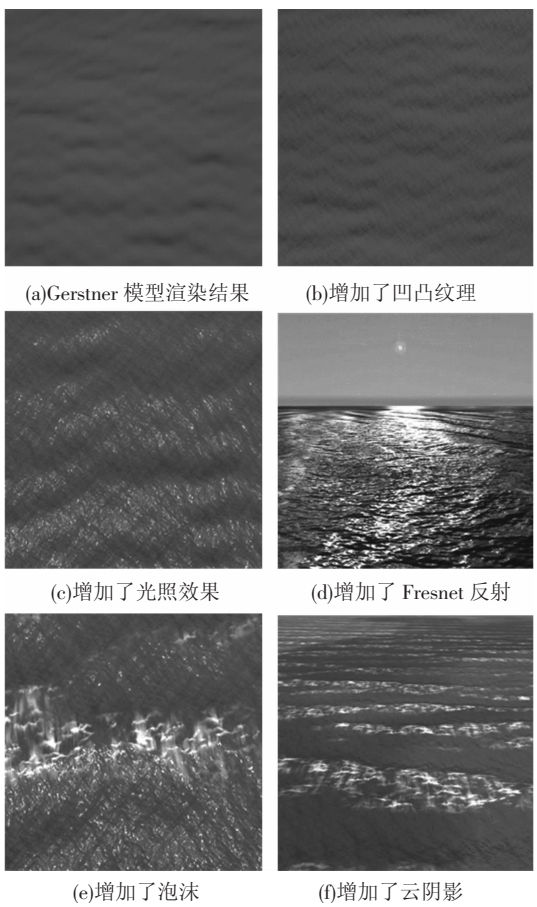
Fresnel 反射可以与天空反射结合在一起考虑,渲染效果如图 2(d)所示.

3.2 泡沫

Choppy 波导致了海平面产生泡沫,泡沫主要出现在波峰附近,对泡沫的模拟可以增加模拟的真实感. 本文采用动画泡沫纹理对其方法进行了改进,并可以交互实现泡沫的产生和消失,渲染效果如图 2(e)所示.

3.3 云的阴影

天空中的物体在太阳的照射下会在海平面投下阴影,对阴影效果的模拟同样有助于提高渲染效果的真实感,本文对天空中的云生成的阴影进行了模拟,云的阴影生成可通过阴影算法得到一张阴影纹理,渲染效果如图 2(f)所示.



(a)Gerstner 模型渲染结果 (b)增加了凹凸纹理

(c)增加了光照效果 (d)增加了 Fresnet 反射

(e)增加了泡沫 (f)增加了云阴影

图 2 海平面的渲染结果

4 仿真实验

本文选取顶点纹理位移法、基于 GPU 的多尺度 FFT 法与本文的方法进行绘制速度对比实验,实验采用普通 PC(Pentium D 3.0 GHz,1 GB 内存,NVIDIA GTX260 + 显卡),在只绘制海面及光照效果时,3 种方法的平均渲染帧率分别为 53、69、400 帧/s,如表 1 所示. 将本文的方法海浪范围

由 10 km 增加到 90 km 时,平均渲染帧率为 380 帧/s,帧率基本保持不变. 实验表明,本文的方法在 3 种方法中渲染速度是最快的,而真实感也是 3 种方法中最好的. 这主要得益于本文采用的 Gerstner 模型和本文采用的网格模型以及动画凹凸纹理.

表 1　不同方法渲染帧率对比　　帧/s

方法	顶点纹理位移	多尺度 FFT	本文
帧率	53	69	400

本文已在某直升机模拟器中用此方法成功的实现了大范围动态海浪的渲染. 在上述实验用 PC 上,包括全球数据库、多种三维实体及多种特殊效果等的复杂场景下,在 1400×1050 的较高分辨率下稳定的运行在 60 帧/s 交互帧率下,渲染效果如图 3 所示.

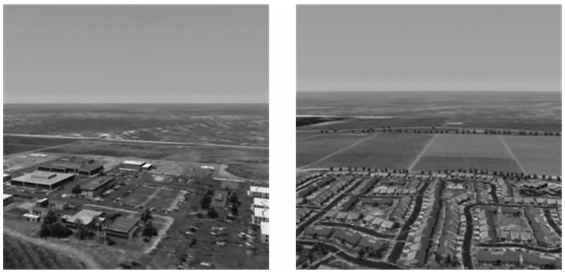


图 3　大范围动态海浪渲染结果

5　结　论

1)提出了细分圆形网格与同心圆相结合的基于 LOD 的网格模型,采用凹凸纹理贴图对距离视点较远处的静态网格渲染,大大提高了渲染速度.

2)基于 GPU 技术实现了基于 Gerstner 模型的海浪运动计算及 choppy 波模拟,并通过动画凹凸纹理改善了渲染效果.

3)模拟了海面的光照、Fresnel 反射、泡沫和阴影等特殊效果. 仿真实验表明,本文提出的研究方法能够对 90 km 的大范围复杂场景下的动态海浪实时渲染.

参考文献:

[1] MIHALEF V, METAXAS D, SUSSMAN M. Animation and control of breaking waves[C]//Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Switzerland: Eurographics Association Aire-la-Ville, 2004: 315 – 324.

[2] CRANE K, LLAMAS I, TARIQ S. Real-time Simulation and Rending of 3D Fluids[M]//NGUYEN H. GPU Gems 3. Boston: Addison-Wesley, 2007: 633 – 675.

[3] STAM J. Stable fluids[C]//Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques. New York: ACM, 1999: 121 – 128.

[4] ENRIGHT D, MARSCHNER S, FEDKIW R. Animation and rendering of complex water surfaces[C]//Proceedings of the 29th Annual Conference On Computer Graphics And Interactive Techniques. NY, New York: ACM, 2002: 736 – 744.

[5] FOURNIER A, REEVES W T. A simple model of coean waves [J]. ACM SIGGRAPH Computer Graphics, 1986, 20(4): 75 – 84.

[6] PEACHEY D R. Modeling waves and surf[J]. ACM SIGGRAPH Computer Graphics, 1986, 20(4): 65 – 74.

[7] FRÉCHOT J. Realistic simulation of ocean surface using wave spectra[C]//Proceedings of the First International Conference on Computer Graphics Theory and Applications. Setúbal: GRAPP, 2006: 76 – 83.

[8] TESSONDORF J. Simulating ocean water[C]//Proceedings of the 29th Annual Conference On Computer Graphics And Interactive Techniques. NY, New York, 2001: 1 – 18.

[9] 任鸿翔,尹勇,金一丞. 大规模海浪场景的真实感绘制[J]. 计算机辅助设计与图形学学报, 2008, 20(12): 1617 – 1622.

[10] MICHELL J L. Real-time Synthesis And Rendering of Ocean Water[R]. Marlborough: Array Technology Industry Technologies Inc, 2005.

[11] KRYACHKO Y. Using Vertex Texture Displacement For Realistic Water Rendering[M]//PHARR M. GPU Gems 2. Boston: Addison-Wesley, 2005: 283 – 294.

[12] FINCH M. Effective Water Simulation From Physical Models[M]//FERNANDO R. GPU Gems. Boston: Addison-Wesley, 2004: 1 – 16.

[13] ISIDORO J, VLACHOS A, BRENNAN C. Rendering ocean water[M]//ENGEL W F. ShaderX: Vertex and Pixel Shader Tips and Tricks. Minesota: Wordware Publishing, 2002: 347 – 356.

[14] 李起成,陈昊罡,汪国平. 动态天空环境下的实时海浪渲染[J]. 计算机辅助设计与图形学学报, 2007, 19 (2): 172 – 177.

[15] BRUNETON E, NEYRET F, HOLZSCHUCH N. Real-time realistic ocean lighting using seamless transitions from geometry to BRDF[J]. Computer Graphics Forum, 2010, 19(2): 487 – 496.

[16] HINSINGER D, NEYRET F, CANI M P. Interactive animation of ocean waves[C]//Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium On Computer Animation. NY, New York: ACM, 2002: 161 – 166.

[17] JOHANSON C. Real-time Water Rendering: Introducing The Projected Grid Concept[D]. Lund: Lund University, 2004.

[18] WLOKA M. Batch, batch, batch: What does it really mean? [EB/OL]. http://developer. nvidia. com/docs/IO/8230/BatchBatchBatch. pdf, 2003/2011 – 2 – 28.

(编辑　张　红)